Constructing Robust Graphs for Community Detection

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a framework called *Locally Boosted Graph Aggregation* for aggregating multiple noisy networks into a single network so as to improve the quality of community detection algorithms on the result. LBGA addresses the problem of finding a single network that faithfully and robustly represents multiple noisy, complementary underlying data sources. We define a new random graph model to model such scenarios in community detection called the *local stochastic block model* (LSBM), and we exhibit the utility of LBGA on this synthetic model as well as real data sets. LBGA outperforms existing network aggregation algorithms when ground truth is available, and it produces high-quality representations of real networks. We argue that LBGA is generic and can be adapted to other application domains.

1 Introduction

Community detection methods in machine learning literature generally assume a single truthful input graph. However, in practical scenarios the data that comprise a network come from multiple sources which may be noisy and may disagree ([16]). For example, in a social network one may communicate with friends via Instagram and family via Facebook. The best way to aggregate this information is unclear, and the choice of representation heavily impacts the performance of subsequent data mining algorithms ([13, 6, 19]). Though the impact of graph representation on subsequent analysis has been studied, few techniques exist for learning conducive graph representations. Aggregation is often ad-hoc in practice, making it difficult to compare algorithms within the same domain using different data sources.

In this paper, we study the problem of constructing a single graph representation that accurately reflects the underlying network structure and allows for better detection of communities scattered across different data sources. We present an aggregation framework called *Locally Boosted Graph Aggregation (LBGA)* which simulates an iterative reward system inspired by boosting and bandit learning. LBGA evaluates the quality of edges locally, so that it can choose aggregations which most accurately represent the local structure of communities observed in real networks ([1, 16]). LBGA relies on the pair of a simple clustering algorithm and a local heuristic quality measure as a proxy for evaluating the quality of intermediate results. We empirically show that our algorithm constructs robust aggregated graph representations for community detection by testing it on synthetic and real-world data sets and comparing it to existing aggregation algorithms.

We emphasize that while this paper specifically addresses community detection, the edge reward mechanism and the graph aggregation steps of LBGA are application agnostic. LBGA can be repurposed for graph aggregation with respect to other applications, a direction we leave for future work. The paper is organized as follows. In Section 2 we review related literature. In Section 3 we discuss in detail the LBGA framework. In Section 4 we present the experimental analysis and results, and in Section 5 we discuss future work.

2 Related work

2.1 Graph representation learning and clustering

Our work generally falls under representation learning for graphs, which includes modeling decisions about the nodes and edges of the graph. Rossi et al. ([26]) taxonomize this field, and show that transformations to heterogeneous graphs can improve the quality of a learning algorithm. Within their taxonomy our work falls under link re-weighting, which includes the work of ([33, 14]). Our setting deviates from these works by allowing different edge types between the same pair of vertices. Also, our approach is stochastic, which we find necessary for learning a robust representation.

([31]) develops a cross-diffusion based fusion framework called SNF. Both SNF and LBGA emphasize local similarities versus global ones. SNF's framework of iterative re-weighting of edges based on message-passing is similar to LBGA. The LBGA framework is different because the re-weighting is based on techniques from boosting, and unlike SNF our algorithm does not rely on consistency across the input graphs. In Section 4.5 we demonstrate empirically how LBGA outperforms SNF on all of our data sets.

Clustering in multilayer networks ([23, 30, 29, 20, 3, 15, 28]) also has close connections to our work. However, the literature does not address scenarios where the information provided by the different sources is complementary or the overlap is scarce. Our approach iteratively selects those edge sources that lead to better clustering quality, independently of disagreement across the different features. Also, these approaches differ fundamentally from LBGA in that they do not produce a graph, which could be used for other purposes. As an example of multi-edge clustering algorithms, we consider the GraphFuse algorithm ([23]) that falls under the category of tensor-based clustering. GraphFuse computes the clustering based on the CP decomposition of the tensor formed by appending the adjacency matrices of the different graph sources. In Section 4.5 we demonstrate that LBGA out-performs GraphFuse in terms of recovering the ground truth clustering across our datasets. ([25, 7]) present approaches for identifying the right graph aggregation, given a complete ground truth clustering or a portion of it. Our framework requires no such knowledge, but we do use ground truth to validate our experiments on synthetic data (Section 4.3).

Most recently Liu et al. [18] proposed a method for simultaneously discovering a graph and constructing a high quality clustering. Their setting is fundamentally different from ours in that we have full access to potentially noisy graphs, and they have partial informtion of a single noise-free graph. While our work is similar in that we both simultaneously leverage local information while producing a good global clustering, our techniques differ fundamentally. Their approach explores greedily while we adapt techniques from bandit learning and boosting to balance exploration with exploitation. Moreover, our techniques do not apply to their setting because they can only "discover" a node once.

2.2 Boosting and bandits

Our framework departs from previous work primarily through its algorithmic inspirations, namely boosting ([27]) and bandit learning ([4]). In boosting, one assumes the existence of a *weak classifier* whose performance is slightly better than random. In a landmark paper ([27]), Schapire showed how to combine weak classifiers into a PAC-learner by a majority voting scheme. One can consider different graph data sources as weak learners, and ask whether one can "boost" them to a good graph. Unfortunately, our setting does not allow pure boosting because boosting requires access to ground truth labels. Even with good input, a community has no known universal measure of quality.

In bandit learning an algorithm receives rewards as it explores a set of actions, and the goal is to minimize a notion of regret. The basic model has many variants, but two central ones are expert advice and adversaries. Experts are functions suggesting what action to take in each round, and regret is measured with respect to the best expert. The adversarial setting involves an omniscient adversary who sets the experts and rewards so as to maximize regret. We can imagine graphs as adversarial experts, and adapt bandit learning techniques to compensate. Indeed, LBGA is a reward system based on the given application and uses update techniques from bandit learning to learn a graph representation. In our setting we only care if the aggregate graph is good at the end, while bandit learning often seeks to maximize cumulative rewards during learning. There are bandit settings that only care about the final result ([5]), but to the best of our knowledge they do not apply to our problem.

The primary technique we use is the Multiplicative Weights Update Algorithm (MWUA). See ([2]) for an overview and an extensive list of successful applications. The algorithm maintains a weight for each element x_j of a finite set X. In rounds, an element x_i is chosen by sampling proportionally to the weights, a reward $q_{t,i}$ is received, and the weight for x_i is multiplied or divided by $(1 + \varepsilon q_{t,i})$, for some parameter $\varepsilon > 0$. After many rounds, the elements with the highest weight are deemed the best and used for whatever purpose needed. Next, we describe how this algorithm is adapted to graph aggregation.

3 The Locally Boosted Graph Aggregation framework

LBGA can runs multiplicative weights for each edge, forming a candidate graph representation G_t in each round by sampling edges, and computing local rewards on G_t to update the weights for the next round. When G_t converges we produce it as output. The remainder of this section expands the details of this sketch and our specific algorithm implementing it.

3.1 Framework details

Let $H_1, \dots, H_m$ be a set of unweighted, undirected graphs defined on the same vertex set V. We think of each H_i as "expert advice" suggesting for a pair of vertices $u, v \in V$ whether to include edge (u, v) or not. Our algorithm combines the H_i into a single graph G^* suitable for the proposed application. We present LBGA in the context of community detection, noting generalizations. Each round has four parts: producing the aggregate candidate graph G_t, computing a clustering $A(G_t)$ for use in measuring the quality of G_t, computing the local quality of each edge, and updating the weights for the edges. After T rounds we output $G^* = G_T$.

Aggregated Candidate Graph G_t: In each round construct G_t as follows. Maintain a weight $w_{u,v,i}$ for each graph H_i and each edge (u, v) in $H_1 \cup \cdots \cup H_m$. Normalize the set of all weights for an edge $\mathbf{w}_{u,v}$ to a probability distribution over the H_i. For each edge u, v, sample an H_i according to this distribution and include the edge in G_t if it is present in the drawn H_i.

Event $A(G_t)$: After the graph G_t is produced, run a clustering algorithm A on it to produce a clustering $A(G_t)$. In this paper we fix A to be the Walktrap algorithm ([24]), though we have observed the effectiveness of other clustering algorithms as well. In general A can be any event, and in this case we tie it to the application by making it a simple clustering algorithm.

Local quality measure: Define a *local quality measure* $q(G, e, c)$ to be a $[0, 1]$-valued function of a graph G, an edge e of G, and a clustering c of the vertices of G. The quality of (u, v) in G_t is the "reward" for that edge, and it is used to update the weights of each input graph H_i. More precisely, the reward for (u, v) in round t is $q(G_t, (u, v), A(G_t))$.

Update Rule: Update the weights using MWUA as follows. Define two learning rate parameters $\varepsilon > 0, \nu > 0$, with the former being used to update edges from G_t that are present in H_i and the latter for edges not in H_i. In particular, suppose $q_{u,v}$ is the quality of the edge (u, v) in G_t. Then, the update rule is defined as follows:

$$w_{u,v,i} = \begin{cases} w_{u,v,i}(1 + \varepsilon q_{u,v}), & \text{if } (u, v) \in H_i \\ w_{u,v,i}(1 - \nu q_{u,v}), & \text{if } (u, v) \notin H_i. \end{cases}$$

3.2 Quality measures for community detection

We presently describe the quality measure we use for community detection. First we define *edge consistency*, which measures whether an edge has endpoints in the same cluster or across clusters:

$$EC_{u,v} = \begin{cases} 1, & \text{if } c(u) = c(v) \\ -1, & \text{if } c(u) \neq c(v). \end{cases}$$

Algorithm 1 LBGA pseudocode. Note that 1_E denotes the indicator function for the event E.

```
Input: Unweighted graphs $H_1, \dots, H_m$,
  a clustering algorithm $A$,
  a local quality metric $q$,
  three parameters $0 < \varepsilon, \nu, \delta < 1/2$.
Output: A graph $G$.
  Let $\mathbf{w}_{u,v} = \mathbf{1}$ for all $u \neq v \in V$.
  Let $U$ be the edges $E(H_1 \cup \cdots \cup H_m)$.
  Let $G_{\text{learned}} = (V, \varnothing)$.
  while $|U| > 0$ do
    Let $G$ be a copy of $G_{\text{learned}}$.
    for $(u,v) \in U$ do
      Let $p_{u,v} = \frac{\sum_i w_{u,v,i} 1_{\{(u,v) \in H_i\}}}{\sum_i w_{u,v,i}}$.
      Flip a coin with bias $p_{u,v}$.
      If heads, include $(u,v)$ in $G$.
    end for
    Cluster $G$ using $A$
    for $(u,v) \in U$ do
      Set $p = q(G, A(G), (u,v))$.
      for $i = 1, \dots, m$ do
        if $(u,v) \in H_i$ then
          Set $w_{u,v,i} = w_{u,v,i}(1 + \varepsilon p)$.
        else
          Set $w_{u,v,i} = w_{u,v,i}(1 - \nu p)$.
        end if
      end for
      Let $p_{u,v} = \frac{\sum_i w_{u,v,i} 1_{\{(u,v) \in H_i\}}}{\sum_i w_{u,v,i}}$.
      if $p_{u,v} > 1 - \delta$ then
        Add $(u,v)$ to $G_{\text{learned}}$, remove it from $U$.
      end if
      if $p_{u,v} < \delta$ then
        Remove $(u,v)$ from $U$.
      end if
    end for
  end while
  Output $G$.
```

We also define *neighborhood overlap* (NO), which asserts that vertices sharing many neighbors are likely to be in the same community. NO declares the quality of (u,v) to be the (normalized) cardinality of the intersection of the neighborhoods of u and v, namely $NO_{u,v} = \frac{|N(u) \cap N(v)|}{|N(u) \cap N(v)| + log(|V|)}$, where $N(x)$ is the neighborhood of x. The additional log factor normalizes the metric to be consistent across G. Our quality metric, *consistentNO*, combines edge consistency with neighborhood overlap by multiplying the two functions. We have also run experiments using more conventional neighborhood metrics, such as the Dice and Jaccard indices ([10]). ConsistentNO outperforms them by at least 10% in our experiments and for brevity we omit the results.

3.3 Implementation

We give pseudocode for our implementation of LBGA in Algorithm 1. The runtime of LBGA is $O(T(|E|Q(n) + A(n)))$, where $|E|$ is the number of edges, $Q(n)$ is the runtime of evaluating the quality function, $A(n)$ is the runtime of evaluating the event A, and T is the number of rounds. Algorithm 1 improves this by fixing edges whose weights have grown $> 1 - \delta$ or $< \delta$ for a new parameter δ. As LBGA learns, the sampling procedure becomes substantially sublinear in the number of edges. Penalizing non-edges ($\nu > 0$) also improves runtime, and LBGA is stable to minor variations in ε and δ. Moreover, our algorithm empirically scales linearly with the size of the input.

4 Experimental analysis

We describe the datasets used for analysis and provide quantitative results for the performance of LBGA. In all of our experiments LBGA was run with parameters $\varepsilon = \nu = 0.2, \delta = 0.05$, and we found little sensitivity to changes in these parameters.

4.1 Synthetic datasets

Our primary synthetic data model is a generalization of the stochastic block model of ([32]). We construct a probability distribution $G(n_i, p_i, r_i)$ over graphs as follows. Given a number n of vertices and a list of cluster (block) sizes $\mathbf{n} = \{n_1, \dots, n_k\}$ with $n = \sum_i n_i$, partition the n vertices into k blocks $\{b_1, \dots, b_k\}$ with $|b_i| = n_i$. Define k graphs $G_1, \dots, G_k$ and set the probability of an edge occurring in G_i with both endpoints in block b_i to p_i, all others occuring with probability r_i. We call this model the *local stochastic block model* (LSBM). To contrast, we define the *global stochastic block model* (GSBM) by setting the probability of an edge occuring in G_i with endpoints in the same block (any block, not just block b_i) to be p_i, all others with probability r_i. Finally, we include *Erdős-Rényi random graphs* ([12])

Dataset	Parameters
LSBM-1	$m = k = 4, n_i = 125, p_i = 0.2, r_i = 0.05$
LSBM-2	$m = k = 4, n_i = 125, p_i = 0.3, r_i = 0.05$
LSBM-3	$m = 5, k = 4, n_i = 125, p_i = 0.3, r_i = 0.05,$ $i = 1, \ldots, m, p_5 = r_5 = 0.01$
LSBM-4	$m = k = 2, n_i = 250, p_i = 0.0348, r_i = 0.02$
LSBM-5	$m = k = 2, n_i = 250, p_i = 0.0212, r_i = 0.01$
LSBM-6	$m = k = 2, n_i = 250, p_i = 0.0136, r_i = 0.005$
GSBM-4	$m = k = 4, n_i = 125, p_1 = 0.1625,$ $p_2 = 0.125, p_3 = 0.125, p_4 = 0.0875, r_i = 0.05$
GSBM-5	$m = k = 4, n_i = 125, p_1 = 0.15, p_2 = 0.1,$ $p_3 = p_4 = 0.05, r_i = 0.05, i = 1, \ldots, m$
ER only	$m = 4, p_i = r_i = 0.01$
DBLP-1	$n = 1230, m = 3$
DBLP-2	$n = 3090, m = 3$
RMining	$n = 90, m = 6$

Table 1: Description of datasets analyzed. Total number of vertices in each synthetic source graph is $n = 500$. The number of graph sources is m. The number of clusters is k. The number of vertices in cluster i is n_i. The within- and across-cluster edge probabilities for graph source i are p_i and r_i, respectively.

alongside LSBM (e.g., LSBM-3) to capture noise combinations. LSBM-4 through LSBM-6 are models that are at the detectability threshold as shown in [9].

4.2 Real datasets

We use a subset of *DBLP* ([17]), an online database of research in computer science. We use the same datasets as ([23]), in which node are authors and the three edge types are citations, coauthorship, and title similarity. The conferences used for DBLP-1 were STOC+FOCS, AAAI, SIGIR, TODS; for DBLP-2 ICDO, PODS, TKDE, and CACM. We use the manual ground truth labeling of ([23]).[1]

RealityMining ([11]) was a 9-month experiment in 2004 which tracked a group of 90 individuals at MIT via their cell phones. The dataset includes voice calls, bluetooth scan events, cell tower usage, and self-reported friendship and proximity data. We used data between 2004-09-23 19:00:00 and 2005-01-07 18:00:00 (UTC-05:00). Nodes are individuals in the study, and weighted edges correspond to the total duration of voice calls, the total amount of time two individuals used the same cell tower, the total number of bluetooth events, and the results of the friendship/proximity surveys for a total of 6 graphs.

4.3 Validation procedure

We now state how we evaluate the quality of LBGA's output.

Recovery of Inherent Clusters. Since the output of LBGA is a graph, we use the walktrap clustering algorithm to extract communities for analysis. When ground truth communities are available we compare them with LBGA's communities using Normalized Mutual Information ([8]). Otherwise we relate our clusters to known features of the dataset.

Quality of Graph Representation. In addition to producing a good clustering, an ideal graph representation also removes cross-community edges and produces a sparser representation. We use the standard Newman modularity measure ([22]) and conductance ([16]) to measure this. Note that *higher* modularity scores and *lower* conductance scores signify stronger community structure. We note two extreme graph representation cases, the empty graph which is perfectly modular and the union graph which is a trivial aggregation. To signal these cases in our results, we display the *sparsity* of the produced graph G^*, i.e. the fraction of edges in G^* out of the total set of edges in all input graphs.

Recovery of Graph Source Contribution. In our experiments some input graphs contribute more to uncovering the underlying community structure, and should have higher edge weights on average. As such we display the average weights of the input graphs, e.g. in Figure 1.

[1]We recognize issues with this approach, for example that many authors who publish in STOC and AAAI are placed some ground truth community according to an undisclosed and arbitrary method.

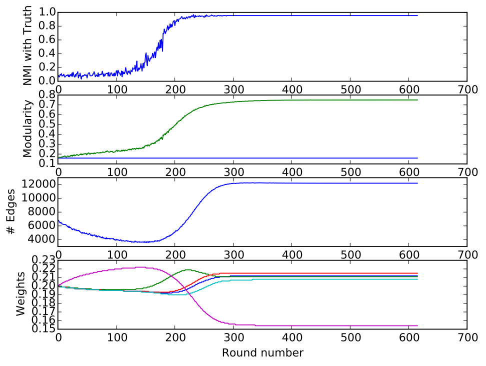

Figure 1: Graph representation learning for LSBM-3. The LBGA parameters are $\varepsilon = \nu = 0.2, \delta = 0.05$. Plots in order top to bottom: 1. NMI of $A(G_t)$ with the ground truth clustering, 2. modularity of G_t w.r.t. $A(G_t)$, with the horizontal line showing the modularity of the union of the input graphs w.r.t. ground truth, 3. the number of edges in G_t, 4. the average probability weight of vertex pairs for H_i. The Erdős-Rényi graph converges to low weight by round 300, even though it is initially favored. Hence LBGA can recover from bad luck and does not boost noise.

Consistency of predicted future links. Given the final learned graph we rank the vertex pairs by their Adamic-Adar score and of the top 100 predicted edges, we report what fraction are within a cluster or across clusters. This data is in Table 3.

4.4 Experimental results

Table 2 states the numerical results of our experiments. As a baseline, we computed the modularity and conductance values of the union of the input graphs with respect to the ground truth (for synthetic) or the Walktrap clusterings (for the real world). For synthetic examples, we compare our results with GraphFuse ([23]) and SNF ([31]) algorithms. Overall, LBGA converges to graphs with high modularity and low conductance. LBGA produces graph representations that induce correct clusterings in almost all cases where ground truth is known, the challenging case being when the noise rate is close to known detectability thresholds.

Synthetic: Figure 1 depicts a run of LBGA with consistentNO on the dataset LSBM-3. LBGA converges quickly to a graph with a perfect clustering and high modularity. We plot the number of edges in G_t over time and the average vertex-pair weight for each input graph. LBGA produces a graph using with 40% sparsity and weights edges from the Erdős-Rényi source appropriately. Our algorithm hence achieves a high quality graph while preserving and highlighting the underlying community structure. Figures 1 and 2 also demonstrate that LBGA does not falsely boost noise to report community structure where none is present. Figure 2 depicts the behavior of LBGA on a dataset of only Erdős-Rényi random graphs. LBGA produces aggregate graphs whose modularity values are low and conductance values are high. We see a clear phase transition in performance around $p = 0.3$ corresponding to the Erdős-Rényi graphs becoming triangle-dense and therefore less distinguishable from graphs that are a single community. Tolerance to such dense levels of noise is unavoidable. Indeed, at the detectability threshold (LSBM-4 through LSBM-6), we achieve NMI that outperforms both the baseline and all the compared methods. For the remaining synthetic models we outperform the compared methods while being comparable with the union baseline and producing a sparser, more modular graph. Indeed, the resulting link prediction scores in Table 3 are far better for LBGA than the union.

DBLP & RealityMining: Table 2 shows LBGA outperforms GraphFuse and SNF with respect to NMI on both DBLP data sets. LBGA also produces a sparse graph of high modularity. The compared methods GraphFuse and SNF do not produce graphs as output, and so we can only compare their NMI. LBGA also performs comparably with the baseline, while producing a sparser, more modular graph. For RealityMining, LBGA's output contains two dense clusters corresponding exactly to the MIT Media Lab and the Sloan Business School, with only three edges crossing the cut. In addition, this graph uses only 63.5% of the total edges available.

4.5 Comparison with GraphFuse and SNF

We compare LBGA with GraphFuse ([23]), a multi-graph clustering algorithm and SNF ([31]), a graph fusion algorithm. We use NMI with the ground truth as the performance measure. For the

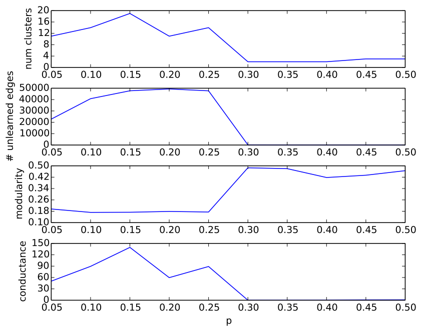

Figure 2: Statistics about the aggregate graph produced by LBGA after 500 rounds on a suite of 4 Erdős-Rényi random graphs on 500 nodes and varying edge probability p.

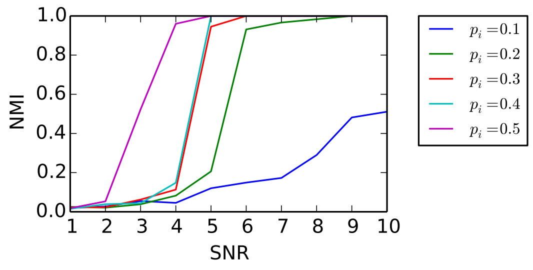

Figure 3: Performance of LBGA (measured by NMI) as a function of SNR for the LSBM model with different probabilities p_i for $consistentNO$.

	Union Graph			GraphFuse	SNF	LBGA: ConsistentNO			
Dataset	Mod.	Cond.	NMI	NMI	NMI	Modularity	Conductance	NMI	Sparsity
LSBM-1	0.103	14.725	0.724	0.686	0.099	0.679 ± 0.114	9.962 ± 23.0	0.503 ± 0.05	0.263 ± 0.04
LSBM-2	0.166	11.233	0.992	0.760	0.180	0.740	0.084	0.992	0.420
LSBM-3	0.166	11.216	1.000	0.779	0.209	0.737	0.104	1.000	0.422
LSBM-4	0.204	117.906	0.028	-	-	0.891	99.219	0.107	0.183
LSBM-5	0.371	113.388	0.087	-	-	0.918	459.00	0.200	0.035
LSBM-6	0.283	119.625	0.072	-	-	0.960	322.00	0.184	0.104
GSBM-4	0.178	10.678	1.000	0.716	0.658	0.739	0.084	1.000	0.433
GSBM-5	0.093	15.368	0.636	0.616	0.436	0.727	2.121	0.619	0.235
ER only	-0.002	24.729	-	-	-	0.193	112.947	-	0.230
DBLP-1	0.698	206.928	0.370	0.30	0.184	0.903	311.682	0.345	0.433
DBLP-2	0.601	573.514	0.217	0.12	0.013	0.926	614.953	0.211	0.492
RMining	0.452	70.314	-	-	-	0.246	0	-	0.646

Table 2: LBGA performance results, compared to GraphFuse, SMF, and a baseline union aggregation. All datasets in this table were run with $consistentNO$ using $\varepsilon = \nu = 0.2, \delta = 0.05$. Union modularity and conductance for real datasets was computed with the walktrap clustering. Values were averaged over 10 trials, and when variances $\sigma^2 > 10^{-4}$ were observed, values are reported with $\pm\sigma$. Note that GraphFuse and SNF **do not produce graphs,** and hence we could not report modularity etc. for these methods.

comparison analysis we have only considered the synthetic datasets where the notion of ground truth is known. Table 2 contains the comparison results.

LBGA outperforms SNF in all cases, and by a particularly large margin on the LSBM model. LBGA also outperforms GraphFuse on both the global block models and the lower-noise local block models LSBM-2 and LSBM-3. LBGA also produces very sparse representations that may be useful for future analysis, while GraphFuse and SNF produce only a clustering. We also note that SNF fails to detect the extreme case of LSBM, where each graph has a clique on a different (disjoint) subset of vertices.

Dataset	LBGA within	union within
LSBM-1	1.00	0.58
LSBM-2	1.00	0.96
LSBM-3	1.00	0.94
LSBM-4	0.91	0.46
LSBM-5	0.08	0.58
LSBM-6	1.00	0.54

Table 3: Link prediction consistency for LBGA versus the baseline union on the LSBM datasets (LSBM-4 through LSBM-6 are at the detectability threhsold for stochastic block models). For each dataset and method the final graph was used to rank vertex pairs for link prediction, and we report the fraction of top-100 ranked links which are between clusters.

4.6 Sensitivity analysis

We analyze the sensitivity of LBGA to noise. In Figure 3 we display NMI for the LSBM model and varying intra-cluster edge probability p_i and varying signal-to-noise ratios. As expected, NMI falls as the noise rate r_i increases. LBGA reaches higher quality and maintains the quality longer for denser graphs, which is also consistent with our expectations. At a signal to noise ratio of 2 or less, the NMI drops to non-useful levels regardless of p_i. The sharp drop in quality is related to well-known phase transitions for community detectability ([21]).

5 Conclusion

LBGA offers a flexible, local aggregation method for combining different graph sources in order to better represent community structure in networks. We derive LBGA from a solid theoretical foundation in boosting and bandit learning, and demonstrated LBGA as a proof of concept on synthetic and real networks. LBGA also simplifies the task of designing a graph aggregation algorithm into utilizing a principled quality measure q and global event A. Doing so allows us to connect the utility of the graph representation to the application of interest.

There are some natural directions to pursue in further studying LBGA. For community detection, we can improve LBGA in a number of ways. Our consistentNO metric is simple, and a more sophisticated metric comparing a local neighborhood to a given null model is likely to provide improvements. Additionally, we use the walktrap clustering algorithm as a black box in the "event" step of LBGA, and walktrap makes some simplifying decisions to arrive at a final clustering. A direction for future work is to use a modified walktrap event that outputs raw similarity values before constructing a clustering, and incorporating this data into the quality measure. Finally, preliminary results of the authors and others show that LBGA can also be improved by incorporating a consensus technique using LBGA as a black box, and that LBGA can detect hierarchical community structure. Further study of these is needed for a better understanding of LBGA.

Another primary direction is to study the utility of LBGA for other data mining techniques, such as link prediction. Since LBGA is modular, one can adapt LBGA to a new application domain simply by defining an event and quality function. A final direction is to prove convergence theorems for LBGA in general and for specific applications.

References

[1] C. Aggarwal, Y. Xie, and P. Yu. Towards community detection in locally heterogeneous networks. In *SDM*, pages 391–402. SIAM, 2011.

[2] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[3] M. Berlingerio, M. Coscia, and F. Giannotti. Finding redundant and complementary communities in multidimensional networks. In *CIKM*, pages 2181–2184, 2011.

[4] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[5] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *ALT*, pages 23–37, 2009.

[6] R. S. Caceres, T. Y. Berger-Wolf, and R. Grossman. Temporal scale of processes in dynamic networks. In *ICDM Workshops*, pages 925–932, 2011.

[7] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *PKDD*, pages 445–452, 2005.

[8] L. Danon, J. Duch, A. Diaz-Guilera, and A. Arenas. Comparing community structure identification. *J. Stat. Mech.*, 2005:P09008, 2005.

[9] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, 84:066106, Dec 2011.

[10] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[11] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[12] P. Erdös and A. Rényi. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[13] L. Getoor and C. P. Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[14] E. Gilbert and K. Karahalios. Predicting tie strength with social media. In *Proc. of SIGCHI Conf. on Human Factors in Computing Systems*, CHI '09, pages 211–220, New York, NY, USA, 2009. ACM.

[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[16] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. (17th) Inter. Conf. on World Wide Web*, WWW '08, pages 695–704. ACM, 2008.

[17] M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *SPIRE*, pages 1–10, 2002.

[18] J. Liu, C. C. Aggarwal, and J. Han. On integrating network and community discovery. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 117–126, 2015.

[19] B. A. Miller and N. Arcolano. Spectral subgraph detection with corrupt observations. In *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2014.

[20] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[21] R. R. Nadakuditi and M. E. J. Newman. Graph spectra and the detectability of community structure in networks. *CoRR*, abs/1205.1813, 2012.

[22] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the US National Academy of Sciences*, 103(23):8577–8696, 2006.

[23] E. E. Papalexakis, L. Akoglu, and D. Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *FUSION*, pages 899–905, 2013.

[24] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[25] M. Rocklin and A. Pinar. On clustering on graphs with multiple edge types. *Internet Mathematics*, 9(1):82–112, 2013.

[26] R. A. Rossi, L. McDowell, D. W. Aha, and J. Neville. Transforming graph data for statistical relational learning. *J. Artif. Intell. Res. (JAIR)*, 45:363–441, 2012.

[27] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

[28] M. Shiga and H. Mamitsuka. A variational bayesian framework for clustering with multiple graphs. *IEEE Trans. Knowl. Data Eng.*, 24(4):577–590, 2012.

[29] L. Tang, X. Wang, and H. Liu. Community detection via heterogeneous interaction analysis. *Data Min. Knowl. Discov.*, 25(1):1–33, 2012.

[30] W. Tang, Z. Lu, and I. S. Dhillon. Clustering with multiple graphs. In *Proc. (2009) IEEE Int. Conf. on Data Mining*, ICDM '09, pages 1016–1021, Washington, DC, USA, 2009. IEEE Computer Society.

[31] B. Wang, A. M. Mezlini, F. Demir, M. Fiume, Z. Tu, M. Brudno, B. Haibe-Kains, and A. Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nat Meth*, 11:333–337, 2014.

[32] Y. Wang and G. Wong. Stochastic Block Models for Directed Graphs. *J. Am. Statistic. Assoc.*, 82(397):8–19, 1987.

[33] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *Proc. (19th) Int. Conf. on World Wide Web*, WWW '10, pages 981–990, New York, NY, USA, 2010. ACM.